Histo-Fetch – On-the-Fly Processing of Gigapixel Whole Slide Images Simplifies and Speeds Neural Network Training

Brendon Lutnick[1], Leema Krishna Murali[2], Brandon Ginley[1], Avi Z. Rosenberg[3], Pinaki Sarder[1,2]

[1]Department of Pathology and Anatomical Sciences, SUNY Buffalo, Buffalo, New York, USA, [2]Department of Biomedical Engineering, SUNY Buffalo, Buffalo, New York, USA, [3]Department of Pathology, Johns Hopkins University School of Medicine, Baltimore, Maryland, USA

Abstract

Background: Training convolutional neural networks using pathology whole slide images (WSIs) is traditionally prefaced by the extraction of a training dataset of image patches. While effective, for large datasets of WSIs, this dataset preparation is inefficient. **Methods:** We created a custom pipeline (histo-fetch) to efficiently extract random patches and labels from pathology WSIs for input to a neural network on-the-fly. We prefetch these patches as needed during network training, avoiding the need for WSI preparation such as chopping/tiling. **Results & Conclusions:** We demonstrate the utility of this pipeline to perform artificial stain transfer and image generation using the popular networks CycleGAN and ProGAN, respectively. For a large WSI dataset, histo-fetch is 98.6% faster to start training and used 7535x less disk space.

Keywords: Convolutional neural network, generative adversarial network, tensorflow, whole slide images

INTRODUCTION

A recent push for digitization in the field of pathology has created exciting opportunities for the application of neural networks in diagnostic, prognostic, and theragnostic applications, beyond what could be accomplished with native histology image formats.[1] Whole slide images (WSIs) are high-resolution scans of tissue sections, they are often gigapixel sized and saved with multi-resolution compression formats.[2] This makes them too large to fit into the memory of hardware (i.e., graphics processing units [GPUs]) typically used to train convolutional neural networks (CNNs).[3] To manage the data volume, WSIs are preprocessed by chopping them into patches which are fed to CNN architectures.[3-9] Traditionally, chopping is performed before training, and predetermined patch locations are saved to the disk as images. While this approach works, it is far from ideal and has limited scalability. Large WSI datasets are storage intensive, and chopping before training duplicates this data locally on the disk. To overcome this constraint, we developed histo-fetch, an innovative input pipeline for training CNNs on WSIs with the popular Tensorflow (tf) library.[10] Histo-fetch samples stochastic patch locations from WSI datasets actively during the network training, executing preprocessing, and common data augmentation operations of this data on the CPU while the GPU simultaneously executes training operations [Figure 1].

METHODS

Histo-fetch first does a presegmentation of the WSIs to identify the tissue regions using morphological image processing. A combination of blurring, thresholding through the Otsu method,[11] and binary erosion identifies tissue and background regions. To maintain efficiency, we utilize the multiresolution decoding ability of WSI formats, using the thumbnail resolution for this presegmentation. The low-resolution tissue mask images are saved to the disk as an extremely space-efficient 2-bit portable network graphics file. We use the Tensorflow data.Dataset class to setup the input pipeline.[10] Specifically, a custom python function generates random coordinates

How to cite this article: Lutnick B, Murali LK, Ginley B, Rosenberg AZ, Sarder P. Histo-fetch – On-the-fly processing of gigapixel whole slide images simplifies and speeds neural network training. J Pathol Inform 2022;13:7.

Available FREE in open access from: http://www.jpathinformatics.org/text.asp?2022/13/1/7/335090

Access this article online

Quick Response Code:

Website:
www.jpathinformatics.org
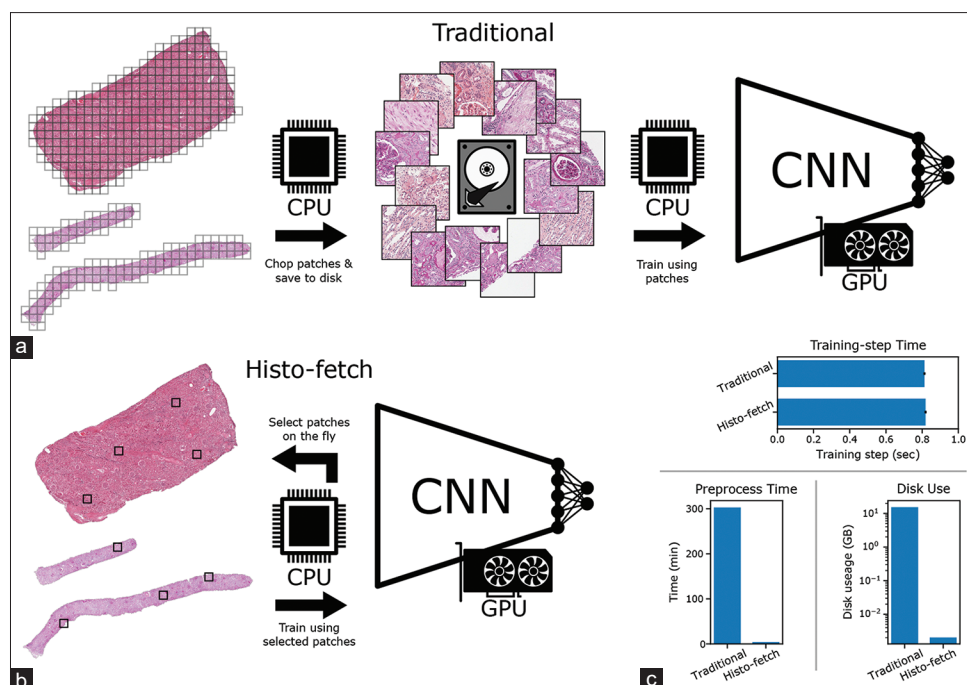
DOI:
10.4103/jpi.jpi_59_20

Figure 1: (a) The traditional method uses the CPU to chop whole slide images into patches which are saved to disk before convolutional neural network training. These patches are read and fed to the graphics processing unit for training. (b) Histo-fetch randomly selects indices containing tissue on the fly. These are processed on the CPU and supplied to the graphics processing unit. (c) Efficiency comparison of the two approaches using ProGAN, highlighting preprocessing time and additional disk space required using a dataset of 151 human biopsy whole slide images. The average training step time does not significantly change.

within the previously identified tissue region, which are then loaded using the OpenSlide-python library.[2] This code takes arguments for the size and downsampling of patches and has the ability to augment the patches through color-shifting, piecewise affine transformation, and rotation/flipping at runtime. For efficiency, we capitalize on the pyramidal multi-resolution encoding of WSIs when training with downsampled image patches, using the optimal slide resolution level for extraction using OpenSlide. This python function wrapped in a Tensorflow py_function operation which allows it to be executed as part of the network graph. Batching the image patches is handled by tf.data.Dataset, whose prefetch operation allows the CPU to preemptively prepare data batches during training.

RESULTS AND DISCUSSION

To demonstrate the usefulness of histo-fetch in training downstream deep learning tasks, we modified two popular generative adversarial networks (GAN) architectures, enabling direct implementation on WSIs. The first of these was the cycle-consistent adversarial network (CycleGAN),[12] a popular architecture for translating between two image datasets. We perform artificial histological stain transfer using our modified CycleGAN, learning to map between stains. This network is extremely simple to train, requiring two WSI datasets (in this case with different stains) to be placed in separate folders. This network used 256 × 256 pixel patches downsampled to ¼ resolution. In-depth details of the training are available in our prior work.[13]

We trained two versions of this network using three different histological stains. Fifty-nine silver and 313 hematoxylin and eosin (H&E) stained WSIs were each used as inputs for the two separate networks which learned to transfer their input to an *in silico* Periodic acid–Schiff (PAS) stain. Fifty-nine PAS WSIs were used by both networks for the training. The slides from all three stains were renal transplant biopsies which were originally obtained for clinical assessment purposes. Examples from these stain transfer networks are visualized in Figure 2a. The generated PAS images preserve the tissue structure of the input. Overall the results look promising. *In silico* silver and H&E images show correctly mapped basement membranes, and tubular brush borders. However, looking closely, the network maps H&E red blood cells into PAS nuclei.

Others have applied CycleGAN for histological stain translation but generally use inefficient WSI chopping before network training.[14,15] de Bel *et al.*[16] in their work mention a random sampling approach seemingly similar to our method; however, they do not provide any details of their algorithm, and their code is not publicly available. We propose that our methodology can be readily used for stain translation across color spaces that are of utility to pathologists in routine practice, without incurring an additional computational expense.

In the second application of our preprocessing method, we adapted NVIDIA's progressive growing of GANs (ProGAN) architecture.[17] To generate realistic looking *in silico* images, this network progressively grows the resolution of
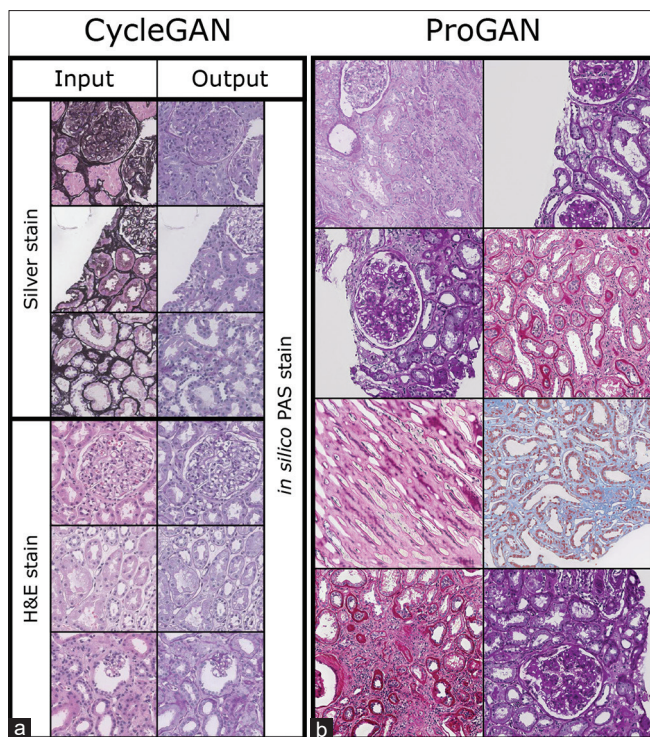
Figure 2: (a) Shows results from two CycleGAN networks, which take hematoxylin and eosin or silver stained input patches and transform them to *in silico* periodic acid–schiff stains. (b) Shows synthetic tissue patches generated using ProGAN trained on 1331 human biopsy (765 GB) whole slide images with various histological stains.

generated images as the training progresses. We trained the modified network up to an image size of 512 × 512 pixels. Macroscopically, the *in silico* images look very realistic, exhibiting natural histological patterns from different stains. Looking closely at the image microanatomy, unexpected morphologies such as fused tubules, thick mesangium, and poorly generated glomeruli and tubule brush borders can be observed. We believe that further training would resolve these issues but this is outside the scope of this paper. Examples of generated image patches are displayed in Figure 2b, with more results available through the link discussed under *code and data availability* section.

Due to the nature of the progressive training, ProGAN requires a multi-resolution dataset. In general, this dataset is prepared by saving training images at all the intermediate resolutions. Instead, we use histo-fetch to extract these patches at different downsampled resolutions as needed during training on-the-fly. The dataset used to train this network contains 1331 WSIs totaling 765 GB disk space of data. This data contained clinically obtained renal biopsies and transplant biopsies stained with PAS, H&E, and Masson's trichrome. Without histo-fetch, training at this large scale would not be possible given the computational resources available typical to standard research labs. This would require extracting all WSI patches and redundantly saving them at resolutions of 4, 8, 16, 32, 64, 128, 256, and 512 square pixels, requiring a massive use of

disk storage. We hypothesize the diverse tissue morphology present in the large WSI training-set directly contributed to the quality of the synthetic images this network generates.

Training with histo-fetch is as simple as passing the folder location of the WSI dataset to the network. To quantify the efficiency of histo-fetch, we chopped a dataset of 151 human biopsies WSIs (18.9 GB) in the traditional method, saving the images to disk. Only regions containing tissue were saved as jpeg images (1200 × 1200 pixels) without overlap. This process took 5 h. 2 min. 57 s. to complete, requiring 15.3 GB of additional disk space. Using more common methods, which save overlapping patches would be even more inefficient and more redundant.[8] In comparison, histo-fetch was 98.6% faster to start training and used ×7535 less disk space [Figure 1c]. From an organizational standpoint, working natively with WSIs greatly streamlines the CNN training workflow using histological data. Files can be managed at the slide level simplifying troubleshooting and data management. While the purpose of histo-fetch is to avoid saving patches, a developer can easily audit the patch selection process by displaying the patches selected at each training step using a python library such as matplotlib,[18] verifying the tiles are selected in a random and appropriate manner. After training is completed, histo-fetch is easily modified to extract predetermined patch locations for prediction on holdout WSIs.

While the scope of this paper focuses on quantifying speed and space utilization improvements of histo-fetch on unsupervised deep learning tasks, in a parallel work we adapted histo-fetch for a supervised segmentation task using the DeepLab v3+ network architecture.[19] This was done by creating additional criteria for the selection of patches, allowing the selection of patches that contain a specific structure at training time. This allows easy class balancing of training data, augmenting regions that occur less frequently to ensure they are seen with sufficient frequency. The details of this implementation and the source code are available in a preprint version of this work.[20]

One benefit of histo-fetch is the random selection of training patches. It is well known that random shuffling of data batches is beneficial during neural network training.[21] We argue that histo-fetch provides a more stochastic delivery of training data than prechopping WSIs. Without predetermined data patch locations, the selection of patches is continuously variable and therefore the network sees greater data variation during training. This improves upon the popular data augmentation strategy of randomly cropping training images, which has been shown to improve training.[22] As a result of the elimination of a preselected dataset, the notion of the training epoch (one training loop through all the data) is no longer valid, the number of training steps should be specified instead.

Processing patches on the CPU at runtime increases the computational overhead; however, training is GPU rate-limited. On our system, the CPU prefetches image patches faster than

the GPU requests them. Training the two networks presented in Figure 2 with a 10 core Intel Xeon ® Silver 4114 CPU and NVIDIA Quadro RTX 5000 GPU, we found no significant difference in GPU utilization or the execution speed of training steps when compared to using preextracted image patches. We quantified this using the ProGAN network, the average training-step was 0.813 ± 0.005 s when chopping patches and encoding them into the default TFRecord format.[10] After updating ProGAN to use histo-fetch the training-step time was 0.819 ± 0.006 s [Figure 1c]. Despite the 0.006 s per step penalty, our ProGAN trained for 1.6M steps using histo-fetch (without WSI chopping and prep of TFRecord files) would still be much faster.

Conclusion

We have developed a method for on-the-fly extraction and processing of patches from WSIs during neural network training. Using this method does not affect the speed of the training, but greatly reduces the time and diskspace requirements of dataset preparation before training. As an added benefit, our method provides greater randomness of data during training than the traditional method, this is due to the continuous randomized sampling of patches during the training process. Added randomness in the training data is commonly believed to be beneficial for network training.[21] Finally, we believe that once set up, this method is easier to use for novices and experienced data-scientists alike. WSI datasets are intuitively managed at the slide level, and changes do not require repreparing the dataset. We believe that any CNN trained on gigapixel scale WSIs would benefit from the use of this method.

Conflicts of interest

There are no conflicts of interest.

Code and data availability

Histo-fetch and modified GANs codes with trained models are available at:

https://github.com/SarderLab/tf-WSI-dataset-utils

https://github.com/SarderLab/WSI-cycleGAN

https://github.com/SarderLab/WSI-ProGAN

ProGAN generated in silico images are available at:

https://bit.ly/3oOQjHn

References

1. Al-Janabi S, Huisman A, Van Diest PJ. Digital pathology: current status and future perspectives. Histopathology 2012;61:1-9.
2. Goode A, Gilbert B, Harkes J, Jukic D, Satyanarayanan M. OpenSlide: A vendor-neutral software foundation for digital pathology. J Pathol Inform 2013;4:27.
3. Cruz-Roa A, Gilmore H, Basavanhally A, Feldman M, Ganesan S, Shih N, *et al.* High-throughput adaptive sampling for whole-slide histopathology image analysis (HASHI) via convolutional neural networks: Application to invasive breast cancer detection. PLoS One 2018;13:e0196828.
4. Lutnick B, Ginley B, Govind D, McGarry SD, LaViolette PS, Yacoub R, *et al.* An integrated iterative annotation technique for easing neural network training in medical image analysis. Nat Mach Intell 2019;1:112-9.
5. Cruz-Roa A, A. Basavanhally F. González, H. Gilmore, M. Feldman, S. Ganesan, *et al*. Automatic Detection of Invasive Ductal Carcinoma in Whole Slide Images with Convolutional Neural Networks. In Medical Imaging 2014: Digital Pathology. International Society for Optics and Photonics; 2014.
6. Ni H, H. Liu, K. Wang, X. Wang, X. Zhou, Y. Qian. WSI-Net: Branch-Based and Hierarchy-Aware Network for Segmentation and Classification of Breast Histopathological Whole-Slide Images. In International Workshop on Machine Learning in Medical Imaging. Springer; 2019.
7. Folmsbee, J., X. Liu, M. Brandwein-Weber, S. Doyle. Active Deep Learning: Improved Training Efficiency of Convolutional Neural Networks for Tissue Classification in Oral Cavity Cancer. In 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018). IEEE; 2018.
8. Ronneberger O, Fischer P, Brox T. U-net: Convolutional Networks for Biomedical Image Segmentation. In International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer; 2015.
9. Hou L, D. Samaras TM Kurc, Y. Gao, J.E. Davis, J.H. Saltz. Patch-Based Convolutional Neural Network for Whole Slide Tissue Image Classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2016.
10. Abadi M, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, *et al*. Tensorflow: A System for Large-Scale Machine Learning. In 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16); 2016.
11. Otsu N. A Threshold Selection Method from Gray-Level Histograms. Vol. 9. IEEE Transactions on Systems, Man, and Cybernetics; 1979. p. 62-6.
12. Zhu JY, T Park, P. Isola, AA. Efros. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In Proceedings of the IEEE International Conference on Computer Vision; 2017.
13. Murali LK, B. Lutnick, B. Ginley, JE Tomaszewski, P. Sarder. Generative Modeling for Renal Microanatomy. In Medical Imaging 2020: Digital Pathology. International Society for Optics and Photonics; 2020.
14. Shaban MT, C. Baur, N. Navab, S. Albarqouni. Staingan: Stain Style Transfer for Digital Histological Images. In 2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019). IEEE; 2019.
15. Xu Z, CF Moro, B Bozóky, Q Zhang. GAN-Based Virtual re-Staining: A Promising Solution for Whole Slide Image Analysis. arXiv preprint arXiv: 1901.04059; 2019.
16. de Bel T, M. Hermsen, J. Kers, J. van der Laak, G. Litjens. Stain-Transforming Cycle-Consistent Generative Adversarial Networks for Improved Segmentation of Renal Histopathology. In Proceedings of the 2nd International Conference on Medical Imaging with Deep Learning; Proceedings of Machine Learning Research; 2019.
17. Karras T, T. Aila, S. Laine, J. Lehtinen. Progressive Growing of Gans for Improved Quality, Stability, and Variation. arXiv preprint arXiv: 1710.10196; 2017.
18. Hunter JD. Matplotlib: A 2D graphics environment. Comput Sci Eng 2007;9:90-5.
19. Chen LC, Y Zhu, G Papandreou, F Schroff, H. Adam. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In Proceedings of the European Conference on Computer Vision (ECCV); 2018.

20. Lutnick B, D. Manthey, JU Becker, B. Ginley, K. Moos, JE Zuckerman, *et al*. A user-friendly tool for cloud-based whole slide image segmentation, with examples from renal histopathology. bioRxiv 2021: p. 2021.08.16.456524.

21. LeCun YA, L. Bottou, GB Orr, KR. Müller. Efficient Backprop, In Neural Networks: Tricks of the Trade. Springer; 2012. p. 9-48.

22. Takahashi R, Matsubara T, Uehara K. Data Augmentation Using Random Image Cropping and Patching for Deep CNNs. Vol. 30. IEEE Transactions on Circuits and Systems for Video Technology; 2019. p. 2917-31.